Lecture 10

Recall the simple algorithm where we maintain a preflow f throughout the algorithm.

$f : E \to \mathbb{R}$ such that $0 \leq f(e) \leq c(e) \ \forall e \in E$

and $\sum_{e: e \text{ entering } u} f(e) \geq \sum_{e: e \text{ leaving } u} f(e) \quad \forall u \in V - \{s\}$.

So f violates the flow conservation constraint

Instead of finding s-t paths and sending flow along these paths, in preflow-push algorithm, we will push flow along 1 edge at a time.

- push (e, δ) is the main operation here.
- another important operation is relabel(u).

relabel (u) : $\ell(u) = \ell(u) + 1$.

This operation increases u's level by 1.

* Please run this algorithm on the example seen in Lecture 7. The algorithm starts with the preflow f where 16 units is sent along (s, v_1) and 13 units is sent along (s, v_2). What happens next?

Exercise 1. Show that throughout the algorithm f is a preflow.

Exercise 2. Suppose the algorithm terminates and returns f. Show that f is a max-flow.

- first show that f is a flow.
- Show that there is never a steep edge in G_f. A steep edge (a, b) is one where $\ell(a) \geq \ell(b) + 2$.
- Hence conclude that there is never any s-t path in G_f.

Thus if the algorithm terminates and returns f then f is a flow and there is no s-t path in G_f. By max flow - min cut theorem, this means f is a max-flow.

Conclusion: If the algorithm terminates then f is a max-flow.

Question: Does the algorithm terminate?
- We will bound the total number of push operations and relabel operations.
  * This will prove the algorithm terminates.

Bounding the number of relabels.

We will show that $\ell(u) \leq \boxed{\quad\quad}$ for every vertex u.

We will fill this blank soon.

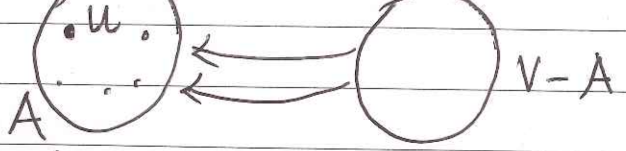
Claim. Any vertex u with positive excess, i.e., any u with $\text{excess}(u) > 0$, has a path to s in G_f.

Assuming the above claim, let us try to fill in the blank. Since there is no steep edge in G_f and it is only vertices with positive excess that get relabelled, the above claim

$\Rightarrow$ $\ell(u) \leq 2n-2$ for every u. (Why?)

Proof of Claim.

Let A be the set of vertices reachable from u in G_f.



Since no vertex in $V-A$ is reachable from u in G_f, all edges between A and $V-A$ in G_f are directed from $V-A$ to A.

Consider the quantity $\sum_{v \in A} excess(v)$.

Date

$$\sum_{v \in A} excess(v) = \sum_{e \in (V-A)\times A} f(e) - \sum_{e \in A \times (V-A)} f(e)$$

Can there be any flow along any edge $e \in (V-A) \times A$?
That is, can $f(e)$ be positive for any $e \in (V-A) \times A$.
- The answer is no since if $f(e) > 0$ for some $e \in (V-A) \times A$ then $e'' \in G_f$ and e'' is directed from A to $(V-A)$. Recall that we observed that G_f has no edge from A to $V-A$.

So $\sum_{v \in A} excess(v) = -\sum_{e \in A \times (V-A)} f(e) \leq 0$.

However $u \in A$ and $excess(u) > 0$.
This means $\sum_{v \in A - \{u\}} excess(v) < 0$.
The only vertex with negative excess is s.
Thus $s \in A$. Hence s is reachable from u in G_f. □

As we saw just before proving this claim, this means $l(u) \leq 2n-2$ for all vertices u.
So any vertex can be relabelled at most $2n-2$ times. Hence the total number of relabels in the algorithm $\leq \underbrace{(n-2)}_{\text{only vertices other than } s,t \text{ can be relabelled.}} \cdot (2n-2) = O(n^2)$

So the preflow-push algorithm can run the "relabel" step $O(n^2)$ times in the entire algorithm.
Whenever the while-loop is run, either "push" or "relabel" happens. So if we show an upper bound on the total number of push operations that can happen, that means this algorithm always terminates.

Bounding the number of pushes

Let us classify push(e, δ) operations into 2 types:

- saturating push : here δ = residual-capacity(e)
- non-saturating push : so $\underbrace{\delta < \text{residual-capacity}(e)}$

this means $\delta = \text{excess}(u)$ where $e = (u, v)$

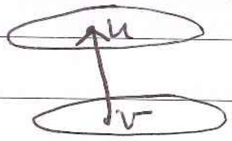
How many saturating pushes can happen along edge e?

Let $e = (u, v)$. Initially $\ell(u) = 0$ and no push can happen along e since there is no vertex at a "Lower level" than u.

- we need 1 relabel (u) operation to have the first saturating push along e.
  - * at this point $\ell(u) \geq 1$

- what about the next saturating push along e?
  - * we need to have $\ell(u) \geq 3$ for this.

This is because after a saturating push along (u, v) :

G_f 

The residual graph G_f has $r(e') = 0$ for the arc $u \to v$

So G_f has only the reverse arc $v \to u$

For (u, v) to reappear in G_f,
we need to have a push along (v, u) — then (u, v) will appear again as the reverse of (v, u).

- for push to happen along (v, u), we need to have $\ell(v) > \ell(u)$. Since $\ell(u) \geq 1$ and level of a vertex can never decrease in the algorithm, we need to have $\ell(v) \geq 2$ for push along (v, u).
- so for the second saturating push along (u, v), we need to have $\ell(u) > \ell(v) \geq 2$, i.e, $\ell(u) \geq 3$.

By the same argument, we have:
- for the i-th saturating push along edge (u,v), $l(u) \geq 2i-1$.

We have shown that $l(u) \leq 2n-2$. This means the total number of saturating pushes along the edge (u,v) is at most $n-1$.

Thus the total number of saturating pushes in the entire algorithm $\leq 2m \cdot (n-1) = O(mn)$

counting the forward arcs & reverse arcs.

We will now bound the total number of non-saturating pushes in the entire algorithm.

This will use a clever argument based on a potential function ϕ. Define $\phi = \sum_{u:\, excess(u)>0} l(u)$

Let us call vertices with positive excess "active".
This function ϕ sums the levels of all active vertices.
So this function changes with time.
Let ϕ_t = value of ϕ at the end of the t-th iteration of the while-loop.

What is ϕ_0? Since all active vertices (these are out-neighbours of s) are in level 0 at the start of the algorithm, $\phi_0 = 0$.

Every iteration of the while-loops performs one of the following 3 steps:
(1) relabel: this increases ϕ by 1.
(2) saturating push: this increases ϕ by at most $2n-3$ since it may activate the end vertex of this edge.

Note that a saturating push may also decrease the function ϕ since it deactivates the start vertex of this edge. All we are saying is that it cannot increase ϕ by more than $2n-3$.

(3) non-saturating push: this operation definitely decreases ϕ by at least 1 since it deactivates the start vertex of this edge.

Since $\phi_0 = 0$, we can write $\phi_t = (\phi_t - \phi_{t-1}) + (\phi_{t-1} - \phi_{t-2}) + \dots + (\phi_1 - \phi_0)$.

So we have $\phi_t = \underbrace{\sum_{i=1}^{t} \Delta\phi_i}$ where $\phi_i = \phi_i - \phi_{i-1}$

let us classify terms here into blue and red

$\Delta\phi_i$ is blue if $\phi_i \geq \phi_{i-1}$ $\rightsquigarrow$ so ϕ did not decrease in the i-th iteration of the while-loop

$\Delta\phi_i$ is red if $\phi_i < \phi_{i-1}$ $\rightsquigarrow$ so ϕ decreased in the i-th itn.

Since $\phi_t \geq 0$, we have sum of all blue terms $\geq$ sum of all red terms.

The sum of all blue terms is at most the total increase in ϕ that can happen and this is $\leq 2n^2$ (total increase due to relabel operations) $+ \underbrace{2mn(2n-3)}_{\text{total incr. due to saturating pushes is at most this quantity}}$

So sum of all blue terms is $O(mn^2)$.

Hence sum of all red terms is also $O(mn^2)$.

Claim. The number of non-saturating pushes in the first t iterations $\leq$ sum of all red terms.

Thus the number of non-saturating pushes in the first t iterations is $O(mn^2)$. This is independent of t, so the number of non-saturating pushes over all iterations is $O(mn^2)$. Hence the while-loop runs for $O(mn^2)$ iterations.